

Facultatea de Automatică și Calculatoare

An I, grupa 30216

Controller de tastatură PS2

### Profesor: Studenți:

Pop Diana Timiș Iulia

Viman Andrei

Cuprins

[Schema bloc a aplicației 6](#_Toc40396946)

[Unitatea de comandă si cea de execuție 6](#_Toc40396947)

[Etapele de proiectare 7](#_Toc40396948)

[Lista componentelor utilizate 7](#_Toc40396949)

[Componenta Manuală 7](#_Toc40396950)

[Componenta Automată 7](#_Toc40396951)

[Componenta **keyboard\_control** 7](#_Toc40396952)

[Componenta **decodificator\_informatie** 11](#_Toc40396953)

[Componenta **control7seg** 13](#_Toc40396954)

[Semnificația notațiilor efectuate și a pinilor interfeței cu exteriorul 15](#_Toc40396955)

[Justificarea soluției alese 16](#_Toc40396956)

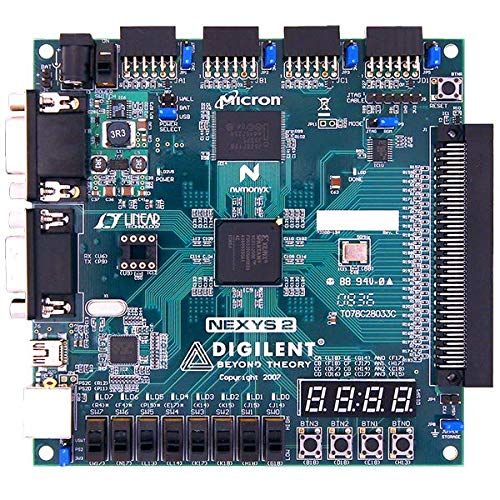
[Instrucțiuni de utilizare și întreținere 16](#_Toc40396957)

[Posibilități de dezvoltare ulterioară 1](#_Toc40396958)8

Specificația proiectului

Să se proiecteze un **controller de tastatură PS2.** Se cere citirea tastelor și afișarea de caractere corespunzătoare pe afișajul cu 7 segmente. Se vor afișa ultimele 4 simboluri, iar tastele de control vor avea roluri speciale (exemplu: Enter începe un rând nou, ștergând afișajul). Poziția punctului pe afișor fiind controlată cu ajutorul săgeților .

Fig. 1. Plăcuța Nexys 2



Protocolul PS2

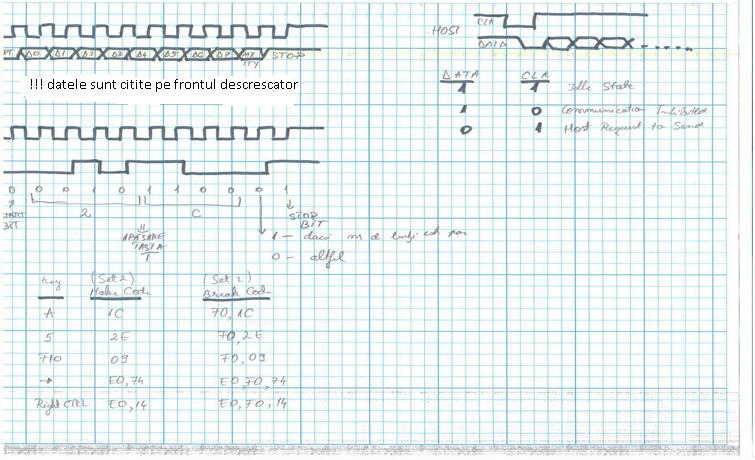
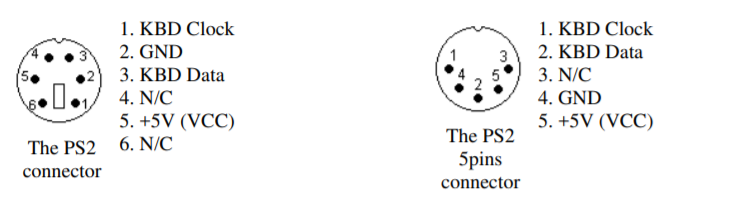


Fig. 2. Protocol și mod de funcționare

Tastatura pe care o aveți cea mai mare parte dintre voi în față, utilizează protocolul standard IBM pentru a comunica cu computerul. Acest protocol are responsabilitatea de a trimite codurile cheie de scanare a tastei pe care o apăsați, la computer și de a obține unele comenzi de răspuns de la tastatură. Aceasta înseamnă că avem de-a face cu un tip bi-direcțional de protocol pe măsură ce fiecare dispozitiv (PC / tastatură) trimite și primește comenzi. La început, vom arunca o privire asupra pinilor mufei conectorului PS2 (mufa poate fi atât 6 pini cât și 5 pini, la tastaturile mai vechi, fiecare din ele funcționând la fel).Putem vedea cum arata cele două mufe în figura de mai jos.

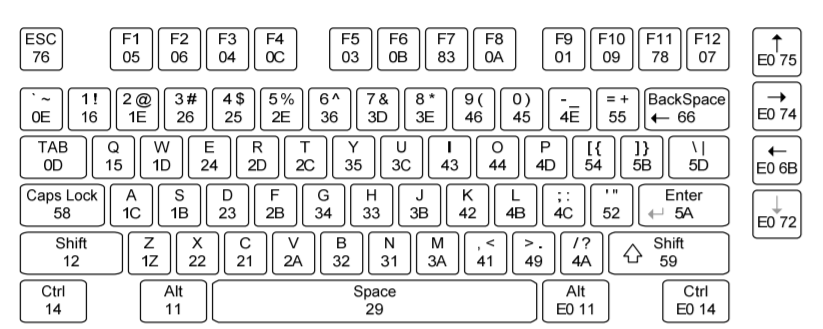


Așa cum am spus anterior, protocolul va realiza comunicarea tastaturii cu dispozitivul la care o conectăm. Acest lucru se realizează folosind datele și liniile de ceas. Aceste linii sunt **unu** atunci când nu are loc nicio comunicare (inactiv). Comunicarea de la tastatură la gazdă și de la gazdă la tastatură sunt ușor diferite, noi vom explica comunicarea tastaturii cu gazda. Rețineți că frecvența de ceas este de aproximativ 25 KHz și este generată de tastatură, dar numai atunci când are loc o transmisie.

Tastatura este liberă să trimită date gazdei atunci când ambele linii , atât cea de date cât și cea de ceas sunt păstrate **unu**. Tastatura va face ca linia de date să se facă zero (bitul de pornire) și apoi va începe să genereze impulsuri de ceas pe linia ceasului. Fiecare bit este trimis în serie cu următoarea ordine: bit de început , mai apoi 8 biți pentru codul tastei urmând ca să transmită mai apoi un bit de paritate iar în final bitul de stop .Fiecare bit este citit pe frontul descrescător al semnalului de tact , astfel încât să fie sincronizat corect așa cum se vede în figura 2.

Ce se întâmplă când apăsăm o tastă? Pur și simplu, tastatura va trimite codul de scanare al tastei pe care ați apăsat-o la gazdă. Tastatura distinge două stări pentru fiecare tastă apăsată: starea de apăsare a tastei și starea de eliberare a tastei. Ce înseamnă asta? Tastatura va trimite codul de scanare pentru tasta pe care ați apăsat-o și când eliberați tasta, va trimite un „F0” urmat de codul de scanare al tastei. O excepție este făcută atunci când apăsăm o tastă specială. Dacă a fost apăsată o tastă cu cod de scanare cu doi octeți (“E0” urmat mai apoi de codul de scanare), la eliberare, tastatura va trimite mai întâi E0, apoi F0 și apoi cealaltă parte a codului de scanare a cheilor. De exemplu, dacă apăsați tasta „A”, codul de scanare 1C va fi trimis. Când eliberați tasta , “F0 1C” va fi trimis. Dacă apăsați tasta „A” și o mențineți apăsată mai mult decât întârzierea normală, tastatura va continua să trimită codul de scanare al tastei în funcție de viteza tipografică, până la eliberarea tastei. Tastele nu au pur și simplu coduri de scanare cu un octet, acest lucru ar putea fi posibil, deoarece un octet poate ține până la 255 de taste diferite. Tastele extinse au doi octeți, iar primul este E0. Cât despre tasta săgeată când o apăsăm va genera “E0 29” urmând ca la eliberare să genereze “E0 F0 29”.

Mai jos puteți observa codurile pentru toate tastele:



# Schema bloc a aplicației

# Unitatea de comandă s i cea de execuție

Pentru proiectul nostru avem un singur mod de lucru , ceea ce înseamnă că **unitatea de comandă** este una simplă. În acest proiect noi luam cheia apăsării unei taste , o decodificăm și o afișăm . Folosim Enter pentru a trece la un rând nou iar săgețile de la tastatură le folosim pentru a ne deplasa pe afișorul cu segmente.

Pentru **unitatea de execuție** avem nevoie de un divizor de frecvență care transformă din 50 MHz în 25MHz și în 10KHz , de un multiplexor , de un decodificator de informație, de un decodor 2 la 4 ,un numărător pe 2 biți si registre pentru memorare unor chei. Din exterior avem nevoie de un buton de reset pentru a putea inițializa componentele si bine înțeles de clock-ul plăcii FPGA dar și de afișor.

# Etapele de proiectare

## Lista componentelor utilizate

### Componenta Manuală

Portul PS2 transmite date serial în momentul apăsării unei taste cu ajutorul clock-ului integrat în tastatură și cu ajutorul unui alt pin pentru șirul de 11 biți transmis secvențial. Noi suntem nevoiți să inițializăm procesul cu întrerupătorul de reset de la placă iar mai apoi putem începe să tastăm taste de la tastatură.

### Componenta Automată

Componenta automată constă dintr-un divizor de frecvență care este un numărător pe 12 biți si un alt numărător pentru selecția anozilor la afișor , pe 2 biți.

Utilizând clock-ul implicit al plăcii FPGA Nexys 2, care are o frecvență de 50MHz trebuie efectuate următoarele calcule astfel încât numărătoarele să numere în timpii necesari. Frecvența de 25 Mhz, care este prima ieșire a numărătorului, și o folosim la debouncer pentru a filtra clock-ul PS2 al tastaturii și intrarea de informație. Semnalul de clock de 10 Mhz îl folosim la numărătorul pe 2 biți care ajută la selecțiile multiplexorului și la selectarea anozilor pentru afișare (clock-ul de 10MHz este a-12 a ieșire a numaratorului pe 12 biți).

=> n=1 (prima ieșire a numaratorului)

=> =5000 => n= 12 (ultima ieșire a numaratorului)

### Componenta **keyboard\_control**

Componenta Keyboard\_control este compusă din 2 componente importante : debouncerul și salvarea cheilor după filtrarea informației.



O imagine care conține text, desen

Descriere generată automat

3.1 Debouncer

Debouncerul ne ajută să filtrăm biții trimiși secvențial de tastatură, deoarece avem de-a face cu date destul de zgomotoase (se poate vedea in figura de mai sus).

Filtrul funcționează cu ajutorul clock-ului de 25 de MHz, și cu un întrerupător de pe placa FPGA (pe post de reset), dar și cu datele transmise de tastatură . Mai avem nevoie de două registre pe 8 biți, cu ajutorul cărora vom filtra datele(regiștrii sunt semnalele PS2c\_filtru și PS2d\_filtru din arhitectură). Registrele shiftează cu câte o poziție la dreapta pe frontul crescător al clock-ului de 25 MHz. Când aceste registre se vor umple ori cu 0 ori cu 1 , se vor genera noi ieșiri (de exemplu dacă PS2c\_filtru este ”11111111” atunci pe PS2c\_f vom pune valoarea 1, iar dacă pe PS2c\_filtru este ”00000000” atunci pe PS2c\_f vom pune valoarea 0; dacă PS2d\_filtru este ”11111111” atunci pe PS2d\_f vom pune valoarea 1, iar dacă pe PS2d\_filtru este ”00000000” atunci pe PS2d\_f vom pune valoarea 0) asemănătoare cu intrările de la tastatura.

3.2 Salvarea datelor în registre

Salvare datelor în valoare1, valoare2, valoare3 (ulterior în key1, key2, key3) se realizează cu utilizarea unei variabile (counter\_max) inițializate cu valoarea 11, cu ajutorul unui numărător (bit\_counter) care se incrementează atunci când PS2C\_f este 1 logic, dar și cu ajutorul unui registru care se concatenează cu PS2d\_f (altfel spus, introducem un bit nou si shiftăm registrul). Atât timp cât bit\_counter este mai mic ca și counter\_max, tot concatenăm regisrtrul cu PS2d\_f. Când se atinge valoarea maximă înseamnă ca s-au transmis (serial) cei 11 biți și este momentul ca valoare1 să ia din cei 11 biți doar 8, deoarece codurile propriu zise ala tastelor sunt pe 8 biți, restul de biți fiind doar de control ( primul bit din cei 11 este mereu 0 urmând cei 8 biți specifici tastei apăsate , urmează mai apoi un bit de paritate iar ultimul bit este întodeauna 1). La sfârșit ,după ce am luat fiecare bit reinițializăm bit\_counter cu 0.

După ce am luat prima valoare transmisă, urmează să luam și cea de-a doua valoare pe care o salvăm într-un mod asemănător. După ce în registru avem cele 11 valori, vom da la registrul valoare2 cei 8 biți , apoi la bit\_counter dăm din nou valoarea 0. Urmează să verificam dacă acești biți din valoare2 sunt egali cu F0 (in hexazecimal). Dacă da, atunci înseamnă că utilizatorul a eliberat tasta și s-a generat codul de eliberare a tastei . Dacă tasta este egală cu E0 (in hexazecimal), atunci înseamnă că avem de-a face cu o tastă specială, care are nevoie de 2 octeți și în acest moment trebuie să luăm cea de-a treia cheie pe care am realizat-o exact la fel cu cele două de mai sus. În final vom pune pe key1, key2, key3 valorile din valoare1, valoare2 si valoare3.

O imagine care conține text

Descriere generată automat

### 4.Componenta **decodificator\_informatie**

Această componentă transformă cei 8 biți aflați pe fiecare dintre registrele key1 și key2, în modul în care, aceștia, să se poată afișa pe afișor-ul plăcii. Acesta conține un ” numărător ” ( ipotetic spus deoarece el se comportă mai mult ca o variabilă din care vom scădea sau aduna 1 în funcție de apăsarea unor taste) pe 2 biți și 4 registre (y0,y1,y2,y3,a\_to\_g) pe 7 biți corespunzători ledurilor afișoarelor.

Această componentă trebuie inițializată cu ajutorul clr-ului de la switch. Astfel, dacă clr este 1, sau dacă biții din key1 sunt asmenea bții-lor generați de apăsarea tastei Enter, atunci pe toți regiștrii intermediari vom pune 7 de 1 (deoarece, afișorul este activ pe 0) și tot aici inițializăm variabila (”contor”) cu 0.

O imagine care conține text

Descriere generată automatUrmătoarele operații se fac doar la eliberarea unei taste (altfel spus, dacă valoarea din key2 este F0) ; în caz negativ, așteptăm până ce ea devine iar un caz pozitiv și vedem dacă tasta apăsată este o săgeată în sus sau la stânga , atunci vom incrementa variabila ”numărător”. Dacă este săgeata în jos sau dreapta vom decrementa variabila .

Această variabilă este importantă (cu ajutorul ei putem realiza un registru de deplasare care deplasează 7 biți simultan) deoarece datorită ei, vom știi de pe care dintre cele 4 afișoare trebuie să proiectăm informația.

În figura de mai jos se pot observa tastele pe care le decodificăm, dar și cum ar arăta acestea pe afișorul cu 7 segmente.

O imagine care conține text

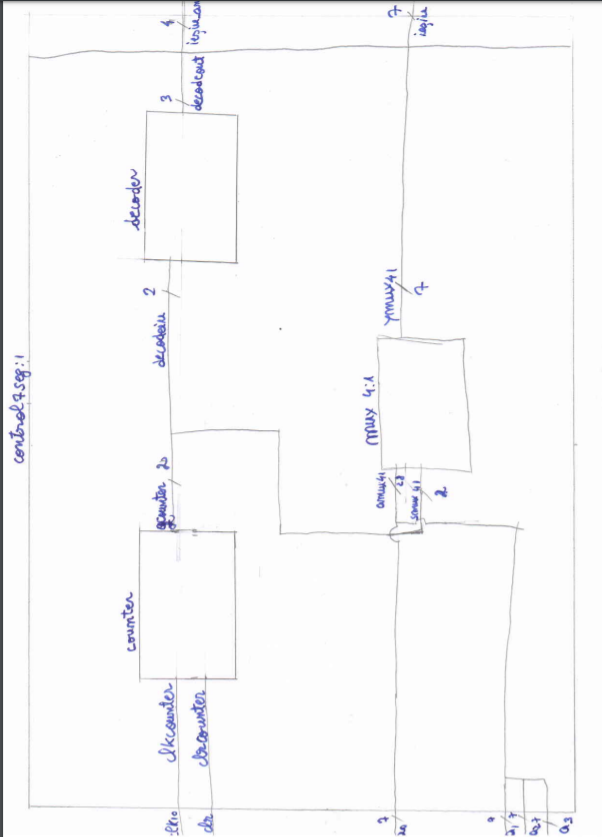
Descriere generată automat

### O imagine care conține text Descriere generată automat Componenta **control7seg**

Această componentă conține alte sub-componente : un multiplexor 4 la 1 (fiecare intrare din cele 4 este pe 7 biți, ieșirea fiind tot pe 7 biți ) , un numărător pe 2 biți, dar și un decodificator 2 la 4. Rolul fiecăruia fiind explicat mai jos.

5.1 Numarătorul pe 2 biti

Acesta numără de la 0 la 3. Avem nevoie de el pentru selecțiile multiplexorului, dar și pentru intrările decodificatorului. Pentru inițializarea lor avem nevoie de un reset . Când resetul este 1 se va face inițialializarea cu 0, iar când resetul este 0, numărătorul va număra crescător. Clock-ul numărătorului este de 10 KHz, acesta numărând pe frontul crescător al lui. De ce un clock cu o frecvență așa de mare ? Urmează să explicăm acest lucru când vom ajunge la decodificator (puțin mai jos ).



5.2 Mltiplexorul 4 la 1

Cu ajutorul multiplexorului, în funcție de selecție, ( ieșirile numărătorului ) vom afișa pe afișor câte una dintre ieșiri pe rând.

Răspunsul la întrebarea ”De ce avem nevoie de un clock de 10 KHz?” este: avem nevoie de el pentru a putea afișa la o viteză suficient de mare, încât ochiul uman să nu sesizeze pauza dintre selecția anozilor . Toate aceste lucruri se datorează construcției plăcii, care are pentru toate cele 4 afișoare, 8 intrări: una pentru punct, una pentru ledul a, una pentru ledul b, una pentru ledul c, una pentru ledul d, una pentru ledul e, una pentru ledul f, una pentru ledul g și alte 4 intrări pentru anozi. Prin comutarea lor cu o viteză suficient de mare, vom reuși să păcălim ochiul uman ca și cum ar afișa pe toate afișoarele, el de fapt afișând doar pe cate unul, pe rând.

5.3 Decodificator 2 la 4

Intrările acestui decodificator sunt ieșirile numărătorului. Ce face acesta?

El reușește să convertească codul format din cei doi biți, în 4 biți convenabili pentru anozii plăcii. Astfel, el din 00 va face 1110, din 01 va face 1101, din 10 va face 1011, iar din 11 va face 0111. Acest lucru se realizează pentru afișarea pe afișor cu 7 segmente.

#### Lista componentelor utilizate:

* Numărătoare
* Multiplexor
* Decodificatoare
* Diverse registre de memorare
* Diverse registre de deplasare
* Componenta keyboard\_control
* Componenta decodificare\_informatie
* Componenta control7seg

### Semnificația notațiilor efectuate și a pinilor interfeței cu exteriorul

Explicația alegerilor făcute în fișierul de constrângeri ufc:

**clk50** (oscilatorul plăcii fpga - ”b8”) este clock-ul propriu-zis al întregului sistem, cu ajutorul căruia se face filtrarea datelor și afișarea lor.

**PS2D** și **PS2C** ( datele și clock-ul de la portul PS2 - ”p11” ”f17”)

**a\_to\_g(6:0)** (ledurile afișoarelor - “l18”, ”f18”, ”d17”, ”d16”, ”g14”, ”j17”, ”h14”) active pe 0, realizează afișarea de cifre si litere.

**anode(3:0)** (anozii afișorului „f17”,”h17”,”c18”,”f15”) selecția anozilor afișorului

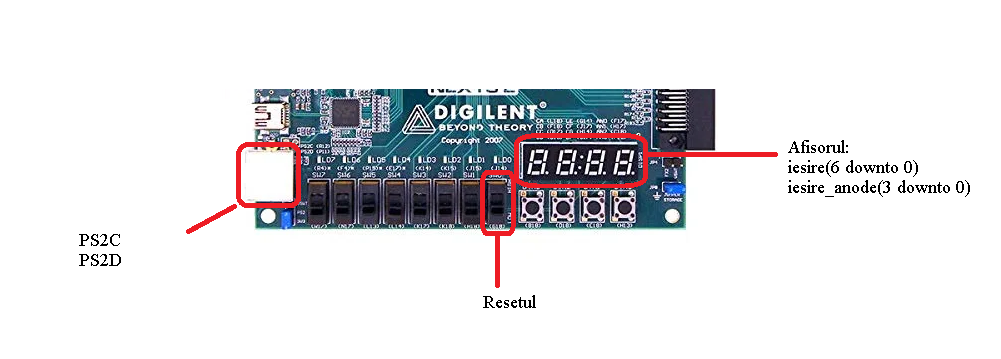
**clr** (SW0 - „g18”) semnalul de ”*reset*” pentru întregul sistem

Fig.5. Semnificația pinilor interfeței cu exteriorul

## Justificarea soluției alese

16

La realizarea acestui proiect, am optat pentru obținerea unui cod lizibil și ușor de înțeles pentru o altă persoană. Pentru ca acest lucru să fie posibil, am utilizat nume semnificative pentru intrări și ieșiri, dar si pentru componentele în sine.

Am evitat obținerea unui cod sursă foarte mare, din această cauză, am separat secvențele de cod pe componente.

Pentru controller-ul PS2, am utilizat componente simple: numărătoare, multiplexor, decodificatoare, cât și elemente puțin mai complicate, spre exemplu componenta keyboard\_control.

Ușurința în utilizare e datorată butonului de restart, care trebuie apăsat o singură dată la pornirea proiectului, apoi noi vom fii nevoiți să apăsam câte o tastă.

Toate în ansamblu reprezintă o soluție eficientă pentru proiectul respectiv.

## Instrucțiuni de utilizare și întreținere

Proiectul dat presupune utilizarea plăcii FPGA *Nexys 2* și respectiv soft-ul *Xilinx ISE Design Suite*.

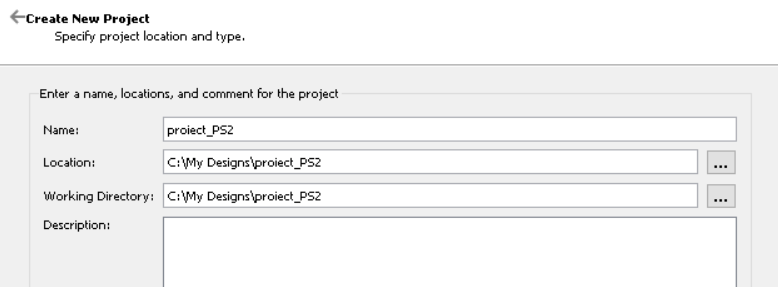
Pentru început, este necesară sintetizarea codului, asignarea pinilor și generarea fișierului cu extensia ”**\*.bit**”, după care este posibilă încărcarea proiectului pe placa FPGA.

Pentru acest lucru vom realiza următorii pași:

1. Lansăm aplicația *Xilinx ISE Design Suite* și creăm un proiect nou:

#### File > New Project

1. Se asociază un nume:



1. Se configurează setările corespunzătoare pentru placa Nexys 2 după cum urmează:

**Family:** Spartan3E

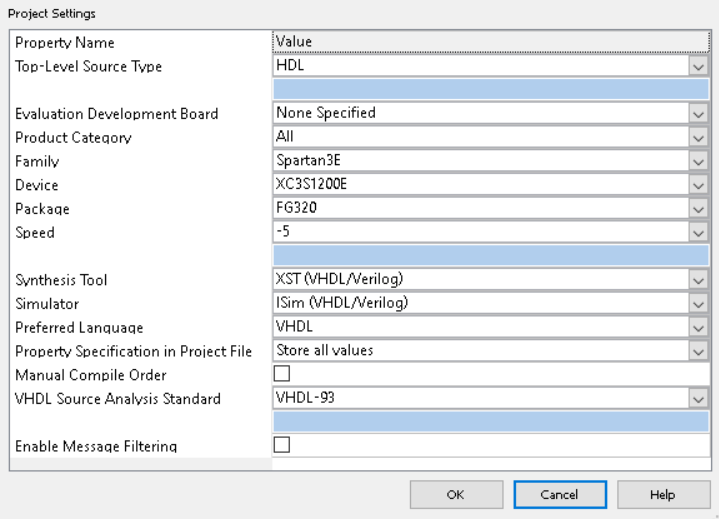
**Device:** XC3S1200E

**Package:** FG320

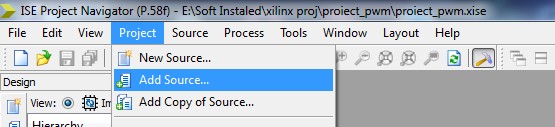
**Synthesis Tool:** XST(VHDL/Verilog)

**Simulator:** ISE Simulator (VHDL/Verilog)

#### Preferred Language: VHDL

**Speed:** -5

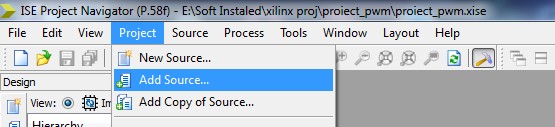
1. Se adaugă fișierul-sursă: **Project > Add Source ...**



Se găsește sursa fișierelor *\*.****vhd*** și se adaugă în program.

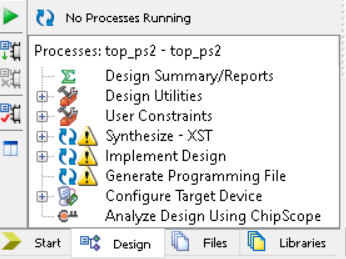
La fel se adaugă și celelalte componente necesare.

1. Se asignează pinii pentru intrările și ieșirile proiectului



Se adaugă în program fișierul constraints.ucf din aceeași locație cu sursa fișierelor \*.vhd

1. Verificăm proiectul și generăm fișierul de execuție

Facem dublu click pe: *Synthesize – XST, Implement Design, Generate Programming File.* Când în stânga acestor trei procese va fi o bifă alba pe fundal verde (ca în imagine) putem trece la următorul pas.

*Configure Target Device –* se lansează programul **Xilinx Impact** (folosit pentru programarea plăcii), se conectează placa FPGA Nexys 2 prin cablu USB la calculator, se accesează comanda **New Project**, apoi **Initialize Chain,** după care se introduce fișierul cu extensia ”\***.bit**” generat de Xilinx și se accesează **Program Device**.

## Posibilități de dezvoltare ulterioară

Proiectul dat poate fi dezvoltat prin adăugarea de noi funcționalități ca:

* 1. Implementarea punctului pe afișor pentru a putea ști pe care dintre cele 4 afișoare ne aflăm.
  2. Implementarea puțin diferită, deoarece , momentan, doar la eliberarea tastei se va afișa ceva pe afișor, iar dacă o să ținem mai mult timp o tastă apăsată , acesta se va scrie tot o singură dată.
  3. Memorarea mai multor valori tastate într-o memorie, deoarece acum sunt păstrate doar 4 valori.